MACHINE LEARNING PROJECT REPORT

Data-Driven HR: Anticipating Job Transitions among Trainees

Group 4A:
Bijay Gautam
Hao-Yun Tseng
Shitong Wu
Suyog Sanklapur
Ting-Hsuan Hsu

List of content

1. Executive summary

Data Scientist, recognized by the Harvard Business Review as the sexiest job in the 21st century, has many job hoppers. Many data scientists are leaving or don't want to have the sexiest job of the 21st century[1]. However, the increasing adoption of Artificial intelligence through the use of big data and data science has increased the demand for data scientists. Although companies can train new candidates to become data scientists, there is still a looming risk that these candidates may leave the company shortly after their training is complete. Job hopping increases costs and wastes valuable company resources. Thus, to reduce unnecessary costs and save valuable company resources, the human resource (HR) department can make a scientific prediction to minimize the risk of hiring a job hopper and maximize the resources for hiring long-term workers. The HR department can apply the models using factors like gender, company_type, and major_discipline to predict the probability of a candidate who will stay in the company vs those who will leave the company shortly after completing their training.

2. Introduction

2.1. The Business Idea

Today, Data Scientists are the pillars helping lead the AI revolution. Thus, to make sure that the company only hires candidates who are serious about staying with the company, we have developed predictive models to help the HR department. With the help of our predictive model, the HR department can forecast if a data scientist is looking for a job change based on their current credentials, demographics, and experience. We will use KNN, Logistics Regression, Decision Tree, and Random Forest. Our analysis focuses on predicting whether the person will look for a new job using personal factors and environmental factors. We have identified the proportion of data scientists actively looking for a job change versus those who are not.

---

[1] Davenport, Thomas H. n.d. "Data Scientist: The Sexiest Job of the 21st Century." Harvard Business Review. Accessed December 4, 2023. https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century.

2.2. Why It's Important

The models will help the HR department become effective and efficient in their hiring practices.

- Cost Savings: Hiring and training new employees is a very costly process. By identifying candidates who are likely to become long-term employees, the HR department can allocate its resources wisely to reduce the cost associated with employee turnover. (also talk about the money that we did not invest in people who might have been interested)
- Refine Hiring and Retention Strategies: Understanding the factors why a data scientist may leave the organization can help the HR department address any issues that current and future employees may face. The HR department can create strategies to target important factors to the data scientist. This can lead to better hiring practices, and attract long-term employees improving employee retention.
- Broader Impact: Although this report will focus mostly on HR Analytics related to Data Scientists, a similar methodology can be used to save cost, and improve hiring practices, and retention rates for other positions. For example, the HR department can apply this methodology when hiring an accountant, an engineer, a banker, a project manager, a sales representative, etc.

2.3. The Data Source

The dataset "HR Analytics: Job Change of Data Scientists", was obtained from Kaggle. Today, Kaggle is the world's largest machine learning and data science community that provides datasets for different cases to help aspiring data scientists learn machine learning techniques.

3. Data Summary

3.1. Data Description

The train data contains 19155 rows and 14 columns. The test data contains 2129 rows and 13 columns. The following list shows the features included in the data set:

- city_ development _index: Development index of the city (scaled)
- gender: The gender of the candidate

- relevent_experience: Relevant experience of the candidate
- enrolled_university: Type of University course enrolled if any
- education_level: Education level of candidate
- major_discipline: Education major discipline of the candidate
- experience: Candidate's total experience in years
- company_size: No of employees in current employer's company
- company_type: Type of current employer
- last_new_job: Difference in years between previous job and current job
- training_hours: training hours completed
- target: 0 – Not looking for a job change, 1 – Looking for a job change

This is the head of the data set and the following is the type of data.

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	major_discipline	experience	company_size	company_type	last_new_job	training_hours	target
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	STEM	>20	NaN	NaN	1	36	1.0
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	STEM	15	50-99	Pvt Ltd	>4	47	0.0
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	STEM	5	NaN	NaN	never	83	0.0
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	Business Degree	<1	NaN	Pvt Ltd	never	52	1.0
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	STEM	>20	50-99	Funded Startup	4	8	0.0

By taking a glance of the data we have, we can tell that most features are categorical (Nominal, Ordinal, Binary). Label encoding is a technique used in machine learning to convert categorical data, which is in the form of labels or text, into numerical values. Since we have so much categorical data, we need to convert them into numerical values first.

Python

```
data =df_concat.copy()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

obj_list = data.select_dtypes(include="object").columns

for obj in obj_list:
    data[obj] = le.fit_transform(data[obj].astype(str))
```

The following graph shows the data type before and after label encoding:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19158 entries, 0 to 19157
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             19158 non-null  int64
 1   city                    19158 non-null  object
 2   city_development_index  19158 non-null  float64
 3   gender                  14650 non-null  object
 4   relevent_experience     19158 non-null  object
 5   enrolled_university     18772 non-null  object
 6   education_level         18698 non-null  object
 7   major_discipline        16345 non-null  object
 8   experience              19093 non-null  object
 9   company_size            13220 non-null  object
 10  company_type            13018 non-null  object
 11  last_new_job            18735 non-null  object
 12  training_hours          19158 non-null  int64
 13  target                  19158 non-null  float64
```

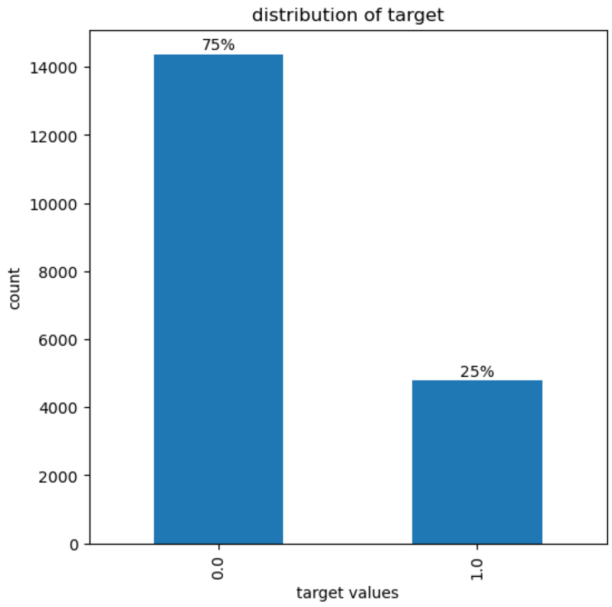
Before

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21287 entries, 0 to 21286
Data columns (total 14 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   enrollee_id             21287 non-null  int64
 1   city                    21287 non-null  int32
 2   city_development_index  21287 non-null  float64
 3   gender                  21287 non-null  int32
 4   relevent_experience     21287 non-null  int32
 5   enrolled_university     21287 non-null  int32
 6   education_level         21287 non-null  int32
 7   major_discipline        21287 non-null  int32
 8   experience              21287 non-null  int32
 9   company_size            21287 non-null  int32
 10  company_type            21287 non-null  int32
 11  last_new_job            21287 non-null  int32
 12  training_hours          21287 non-null  int64
 13  target                  21287 non-null  float64
```

After

3.2. Data Visualization

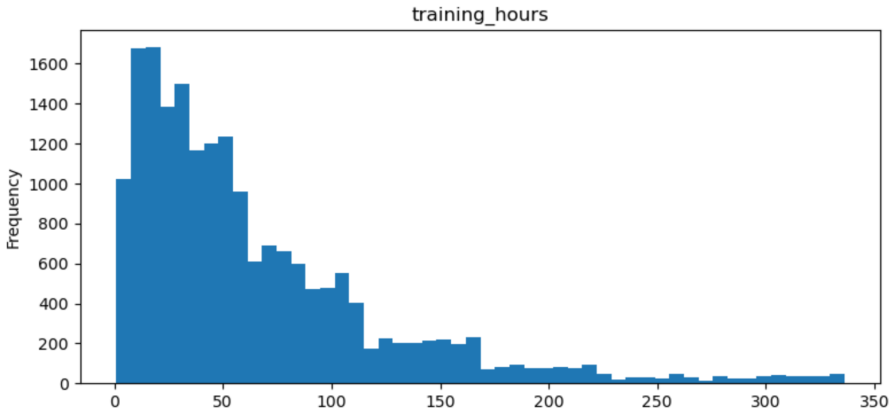
Our prediction target has 2 values: 0 is people who want a job change and 1 represents people who don't want a job change. In our training data set, 75% of the candidates do not want to change their job while only 25% are looking for a job change.



Other features that may relate to their decision of job change can be divided into environmental factors (e.g. training hours, city development index)and personal factors (e.g. gender, years of experience, relevent_experience, education lever, major discipline).

3.2.1. Environmental Factors

- Training hours

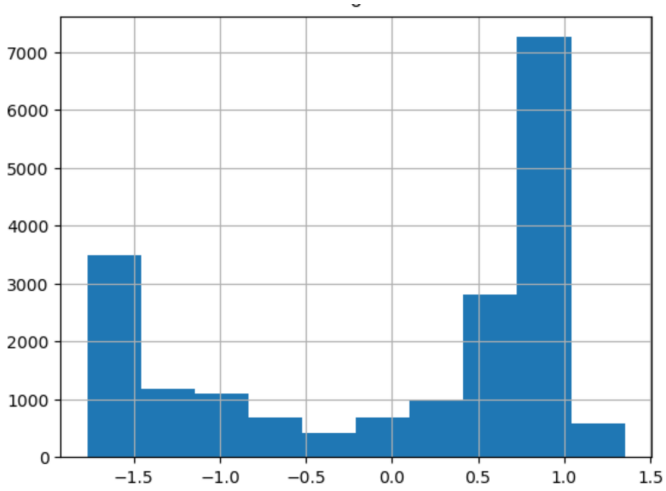


The graph above shows that the training hours are skewed right distribution among trainees. A majority of candidates trained less than 100 hours.

- City development index

Python

```
#Transform 'city_development_index'

from sklearn.preprocessing import PowerTransformer

pt = PowerTransformer(method='box-cox')

pt.fit(df_train['city_development_index'].values.reshape(-1,1))

tr =
pt.transform(df_train['city_development_index'].values.reshape(-1,1))

df_train['city_development_index'] = pd.DataFrame(tr)
pd.DataFrame(tr).hist()
```

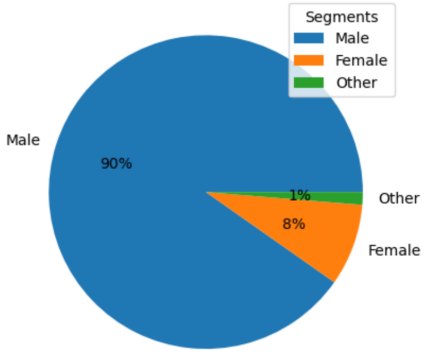


The graph above shows that a majority of people are located in the developed city. However, in the middle of the graph, there is a sudden high in the number of people who live in the less developed city.

3.2.2. Personal Factors

- Gender

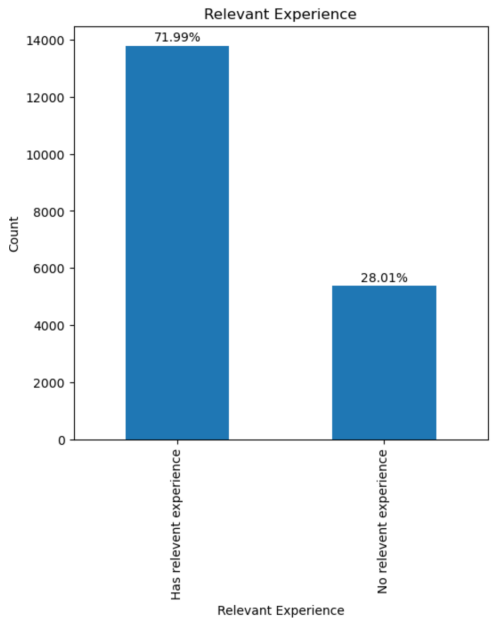
The dataset contains 90% males and 8% females, while 1% identified as other.

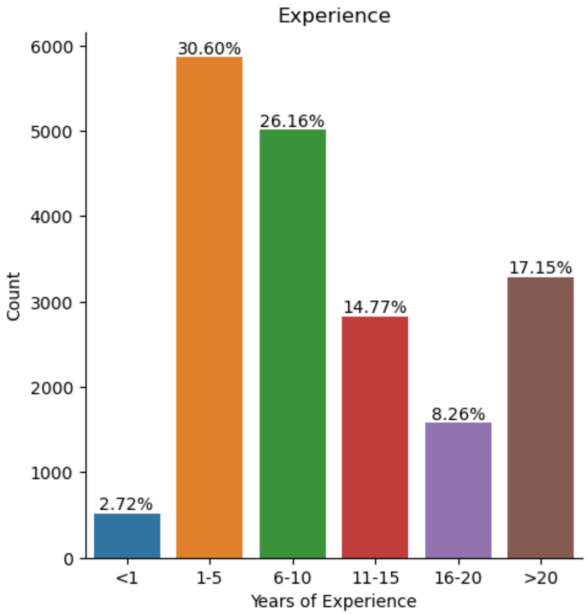


- Experience

The left graph is the representation of the number of people who have relevant data scientist experience. About 71.99% of them have some sort of data scientist work experience, while 28.01% do not.

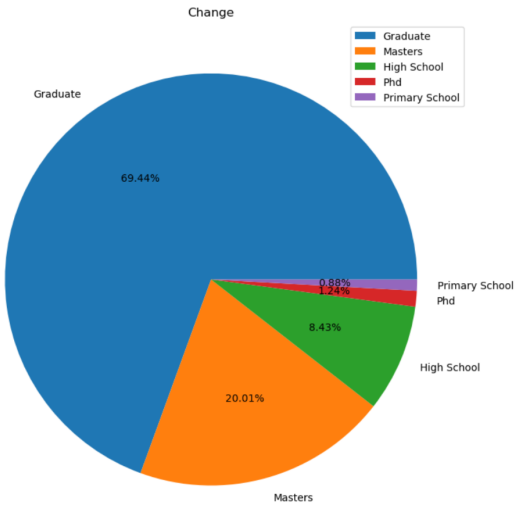
Based on the dataset, about 45.9% have 0-5 years of experience, 26.4% have 6-10 years of experience, 11.3% have 11-15 years of experience, 5.4% have 16-20 years of experience, and about 10.5% have more than 20 years of experience.

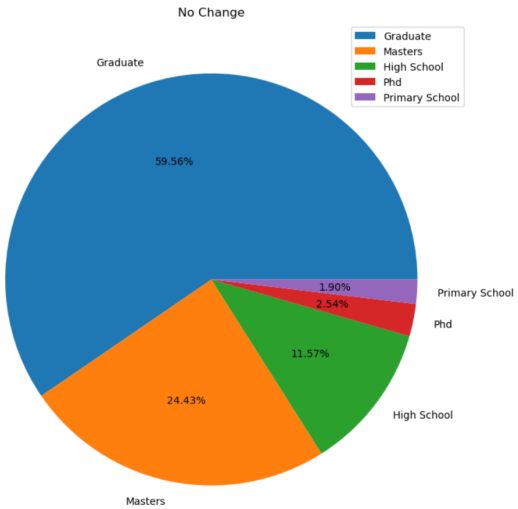




- Education level

Based on the dataset, about 69% have bachelor's degree for people who are seeking a job change, while only 59% have bachelor's degree among those not seeking a job change.

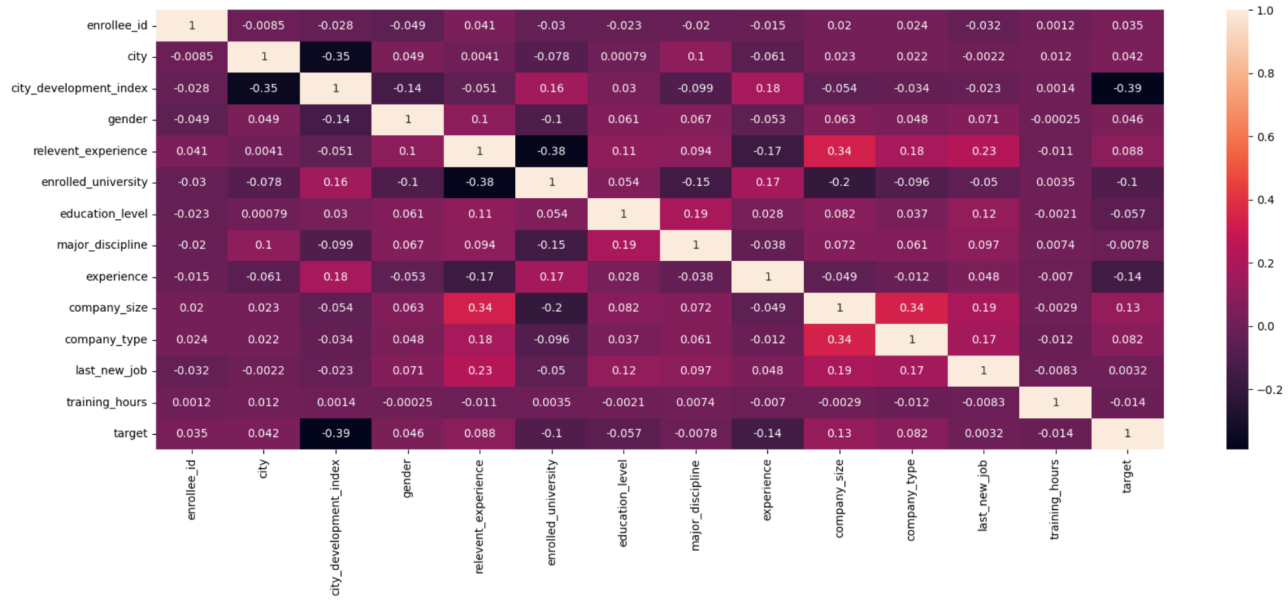




3.3 Data Exploration

3.3.1. Correlation of the data

Correlation is a statistical measure that helps us understand the relationship between different variables in our dataset. The correlation coefficient tells us both the strength and direction of this relationship. A positive coefficient indicates a positive correlation, meaning that as one variable increases, the other tends to increase as well. A negative coefficient indicates a negative correlation, where as one variable increases, the other tends to decrease. By examining these correlations, we gain insights into how variables interact, and this information can be valuable in various fields, from finance to healthcare, and in making predictions using machine learning models.



3.3.2. Benchmark Accuracy

We calculated the benchmark accuracy using some machine learning models. The benchmark accuracy serves as our baseline performance, representing the level of accuracy we would achieve with a basic approach. Our machine learning model's accuracy achieved after data pre-processing can then be compared to this benchmark.

	ACCURACY	PRECISION SCORE	F1 SCORE
KNN	75	16	25
LOGISTIC REGRESSION	77	25	34
DECISION TREE	80	59	60
RANDOM FOREST	79	45	52

The benchmark accuracy we achieved without preprocessing the raw data is pretty high, but we can not simply choose the model with highest accuracy. In a more practical sense, choosing a model should consider the company's specific goals and the trade-offs between false positives and false negatives. What also needs to be considered is the characteristics of our dataset, such as missing data and data imbalance.

4. Data Preprocessing

4.1. Handle Missing Values

Our data set consists of numerous missing values. Handling missing values is a step in preparing data for machine learning models. Inorder to mitigate bias, maintain characteristics of the data, improve overall performance of the models, it is important to handle missing values. Here are several strategies we consider to deal with missing values.

4.1.1. Dropped Missing Values

Dropping rows with missing values is not a feasible option because 46% of the total rows consists of missing values. In addition, this did not improve accuracy, precision and F1 score. This is the percentage of the null values.

	No of null values	Percentage of null values
gender	4508	23.530640
enrolled_university	386	2.014824
education_level	460	2.401086
major_discipline	2813	14.683161
experience	65	0.339284
company_size	5938	30.994885
company_type	6140	32.049274
last_new_job	423	2.207955

As we can see, missing variables were present in a total of 8,953 rows which constituted almost 47% of the total number of rows.

We tried to find out the performance of all models by dropping missing values.These are the findings.

(dropped missing values)	ACCURACY	PRECISION SCORE	F1 SCORE
KNN	83	3	6
LOGISTIC REGRESSION	84	6	10
DECISION TREE	86	47	52
RANDOM FOREST	85	33	42

Though dropping missing values yields high accuracy scores it is not an optimal solution due to the loss of information, reduction in sample size, introduction of bias involved with the process.

4.1.2. Fill with Random Values

In the next step we tried filling up missing values with random values in the rows.This is the observed accuracy,precision and F1 score for each model with this method.

```
Python
#filling missing values in the 'major_discipline' column with random
choices from the existing non-null values

fill_list= df_concat['major_discipline'].dropna()

df_concat['major_discipline'] =
df_concat['major_discipline'].fillna(pd.Series(np.random.choice(fill_lis
t, size= len(df_concat.index))))

sns.kdeplot(x=df_concat['major_discipline'].value_counts())

sns.kdeplot(x=df_before['major_discipline'].value_counts(),color='g')
```

Repeat this method on columns named gender, company_size, company_type.

After this process, we find the percentage of missing values decreased drastically.

```
df_concat.isna().sum()
```

```
enrollee_id              0
city                     0
city_development_index   0
gender                 270
relevent_experience      0
enrolled_university      0
education_level          0
major_discipline       148
experience               0
company_size           335
company_type           337
last_new_job             0
training_hours           0
target                   0
dtype: int64
```

We re-run the four machine learning models and compared the accuracy with benchmark accuracy. Even though the overall accuracy of four models are all good. The precision score and F1 score are much lower.

RANDOM VARIABLES	ACCURACY	PRECISION SCORE	F1 SCORE
KNN	74	7	12
LOGISTIC REGRESSION	76	8	14
DECISION TREE	78	43	49
RANDOM FOREST	77	31	40

4.1.3. Fill with Mean

```
Python
df_mean = df_mean.fillna(train_data.mean())
```

We tried filling up missing values with the mean value of the training data.This is the observed accuracy, precision and F1 score for each model with this method. Due to the high precision score and F1 score, we went on with filling up missing values with mean values.

MEAN	ACCURACY	PRECISION SCORE	F1 SCORE
KNN	74	14	21
LOGISTIC REGRESSION	76	17	26
DECISION TREE	79	54	56
RANDOM FOREST	79	45	51

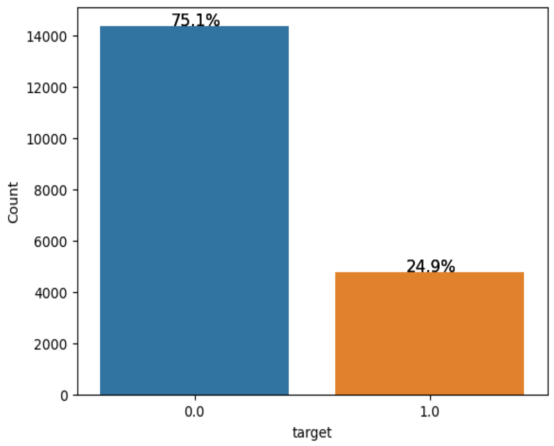
4.2. Handle Imbalance Data

When we analyzed the data set for class distribution, it was highly imbalanced. Before Treating with imbalance data, we first took a look at the stratified accuracy for different classes of our target value.

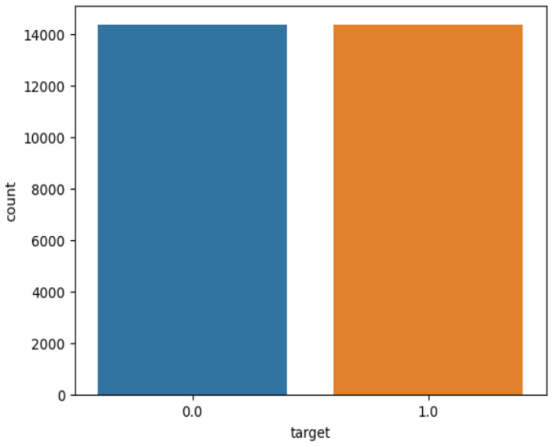
	KNN	Logistic Regression	Decision Tree	Random forest
Accuracy for Class 0	0.94	0.96	0.88	0.90
Accuracy for Class 1	0.14	0.17	0.54	0.45

We find that Accuracy for Class 1 is especially low and may hurt the company if we continue using imbalance data to make predictions, therefore we decided to use SMOTE technique to help in removing this imbalance problem.

```python
from imblearn.over_sampling import SMOTE

smote =SMOTE(sampling_strategy='minority')

x_sm, y_sm = smote.fit_resample(x,y)
y_sm.value_counts()
```



Before SMOTE



After SMOTE

After applying SMOTE, the target data is balanced and ready for data analysis. In addition, we can see that the stratified accuracy for class 1 increased drastically.

	KNN	Logistic Regression	Decision Tree	Random forest
Accuracy for Class 0	0.81	0.74	0.72	0.82
Accuracy for Class 1	0.70	0.74	0.83	0.82

5. Data Analysis

After converting all the attributes to nominal, handeling all the missing values and imbalance data, we run the models and note the overall accuracy obtained. Since this is a binary classification problem, we used the following the four models:

- K-nearest Neighbors (KNN)
- Logistic Regression
- Decision Trees
- Random Forests

When preprocessing the data, we use accuracy score, precision score and F1 score to evaluate the performance of our model. Accuracy is a measure of how often the model makes correct predictions. Precision is the ratio of correctly predicted positive observations to the total predicted positives. F1 score is the harmonic mean of precision and recall. It considers both false positives and false negatives. In addition, an F1 score is particularly useful when there is an uneven class distribution. Since we now get balanced data, we would use accuracy score and precision score to evaluate the performance of our model in the following part.

5.1. K-nearest Neighbors (KNN)

5.1.1. K=4

To begin with, we use KNN algorithm to predict whether the candidate of the training program is looking for a job change. KNN with a lower number of neighbors tends to have a more complex decision boundary. So we start with neighbors=4 and then increase the number of neighbors and compare the accuracy scores.

```python
knnc = KNeighborsClassifier(n_neighbors=4)
knnc.fit(x_train, y_train)
```

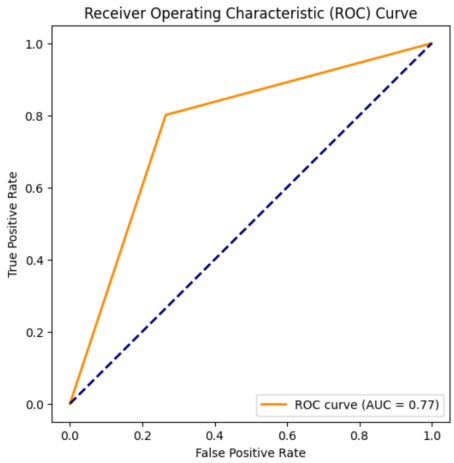
After we run the model, there were 76% correctly predicted values overall and the precision score is relatively high at 0.70.

5.1.2. K=5

Increasing the number of neighbors can make the decision boundary smoother and more robust. It might help reduce the impact of outliers or noisy data points. Therefore, we tried neighbors=5 and re-run the model.

```python
knnc = KNeighborsClassifier(n_neighbors=5)
knnc.fit(x_train, y_train)
```

This time, there were 77% correctly predicted values overall and the precision score is relatively high at 0.80. The AUC score is also reported to be at 0.77.



Compared with KNN(neighbors = 4), we find that increasing the number of neighbors can improve the prediction accuracy. With a small number of neighbors, the model may be overly influenced by individual data points, leading to overfitting. Increasing the number of neighbors

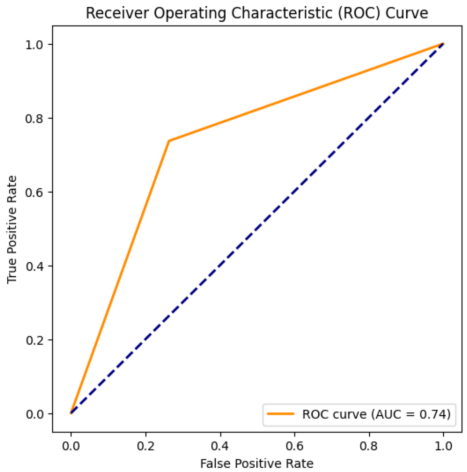
can mitigate overfitting by making the model generalize better to new, unseen data. In addition, with a large dataset that we are analyzing, a higher value of n_neighbors is beneficial for creating a more stable and generalized model. After adjusting the number of neighbors and re-run the KNN model, we finally picked n=5 with higher accuracy.

5.2. Logistic Regression

Firstly, we performed k-fold cross-validation using the logistic regression model. The cross_val_score function evaluates the model's performance for each fold and returns an array of accuracy scores. We found out that the mean accuracy across all folds is approximately 76.22%.

```python
from sklearn.model_selection import KFold , cross_val_score

K_fold = KFold(n_splits=10 , shuffle= True , random_state= 42)

scoring = "accuracy"

score = cross_val_score(lr , x,y, cv= K_fold , scoring=scoring )

print (score)

round(np.mean(score)*100 , 2)

lr.fit(x_train , y_train )
```

Then we start to fit the logistic regression model on the entire training set and report accuracy as well as precision. The reported accuracy score and precision score are both approximately 74%. The ROC is shown as below.



Overall, the logistic regression model achieves a reasonable accuracy on both the training and test sets, and the additional metrics provide a more detailed evaluation of its performance. The cross-validation results help assess the model's robustness across different subsets of the data during training.

5.3. Decision Trees

5.3.1 Individual Decision Tree

```python
dt = DecisionTreeClassifier(max_depth=7)

dt.fit(x_train, y_train)
```

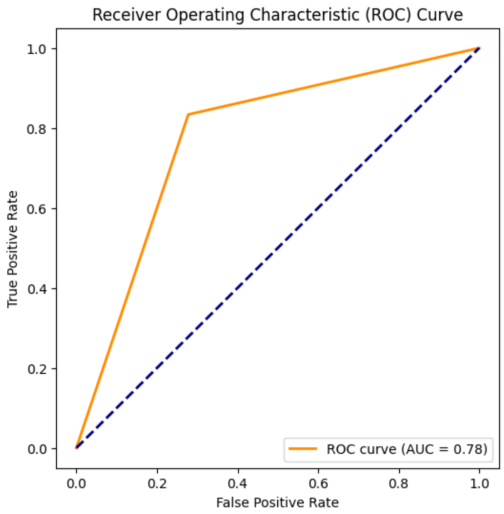
At the beginning, we created a Decision Tree classifier with a specific maximum depth (max_depth=7). After running the individual Decision Tree model, we get an accuracy score of 0.78 and precision score of 0.83.

5.3.2. Grid Search with Cross-Validation

```python
param_grid = {"max_depth": [3,5,7,8,9,10]
,"max_features":[3,5,7,8,9,10]}

dt = DecisionTreeClassifier()

grid_search = GridSearchCV(estimator= dt , param_grid=param_grid , cv =
5)

grid_search.fit(x_train,y_train)

print ("Best: %f using %s" %(grid_search.best_score_,
grid_search.best_params_))
```

In this case, we performed a grid search over a range of hyperparameters (max_depth and max_features). The GridSearchCV method iterates through all possible combinations of hyperparameters, using cross-validation to evaluate the performance of each combination. The best set of hyperparameters is then chosen based on the specified scoring metric (default is accuracy).

This approach is more exhaustive as it searches through a predefined grid of hyperparameter values. It helps to find the optimal combination that performs well on the given dataset.We re-run the Decision Tree model and get an accuracy score of 0.78 and precision score of 0.80, proving that this is a better way to make predictions..



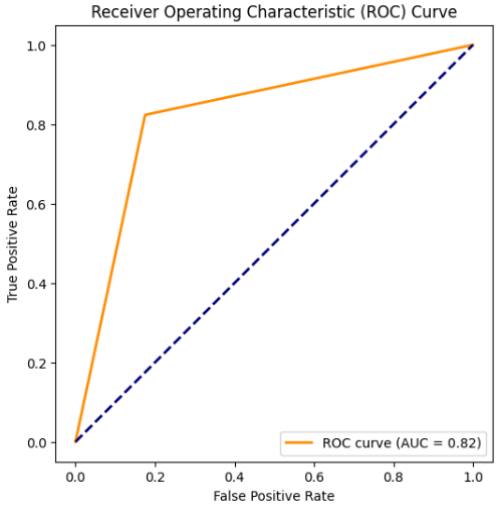
5.4. Random Forests

Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Python

```
from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators = 50, class_weight = {0:1, 1:4})
rfc.fit(x_train, y_train)
```

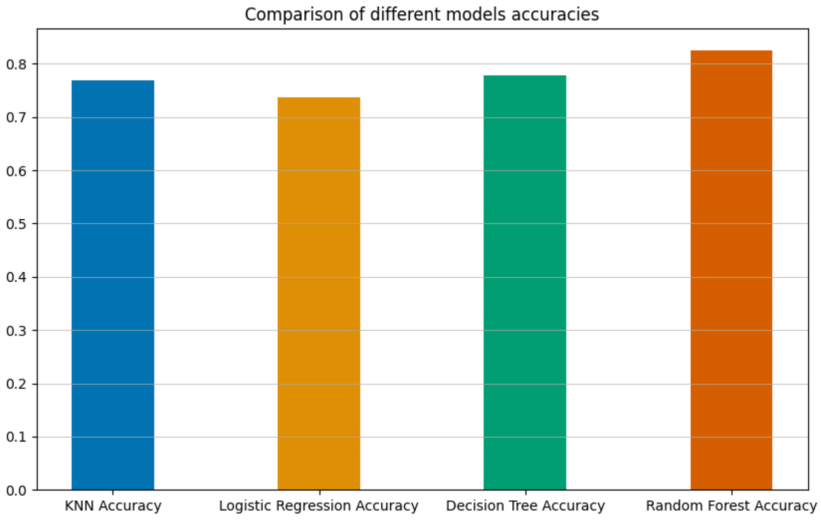
The Random Forest model performs well on the given dataset. The accuracy of 82% suggests that the model is correct in its predictions for about 82% of the instances in the test set. In addition, the precision score is also high, indicating that the model is effective in predicting different classes.



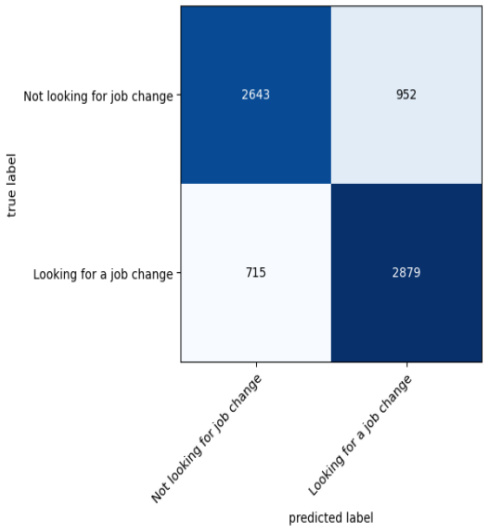
5.5. Choice of Model

5.5.1. Comparison Between Different Models

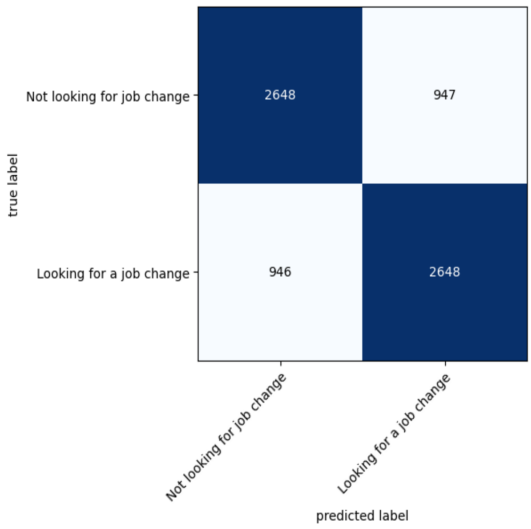
Firstly, we compared the accuracy score achieved by the four models and found out that the random forest model has clearly the highest accuracy.



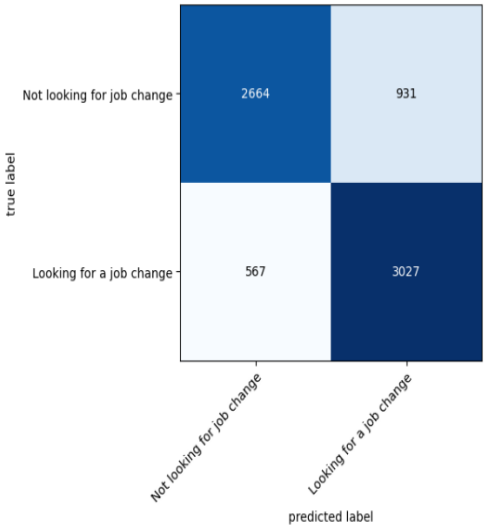
Next, we are going to use confusion metrics for the reason that we can view our results and explain them to business stakeholders more easily.



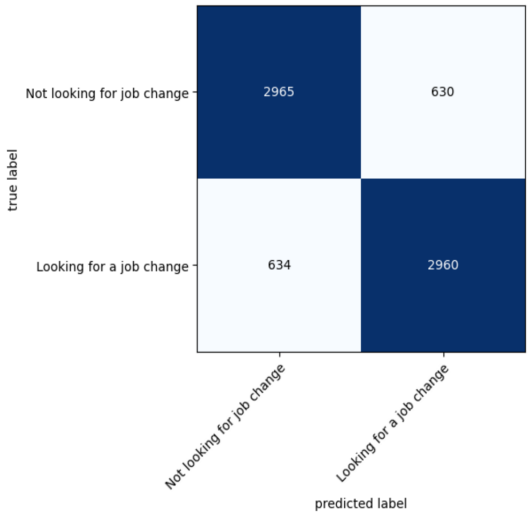
K-nearest Neighbors (K=5)



Logistic Regression



Decision Trees (GridSearchCV)



Random Forests

FP (False Positives): Indicates the instances that were predicted as positive but are actually negative. Having the least number of FP suggests that the Random Forest model is conservative when predicting positive instances, making fewer mistakes by incorrectly labeling negatives as positives.

FN (False Negatives): Indicates the instances that were predicted as negative but are actually positive. Having the least number of FN suggests that the Random Forest model is effective in capturing positive instances, minimizing the instances where it fails to recognize positives.

	FP(False Positives)	FN(False)
K-nearest Neighbors (K=5)	952	715
Logistic Regression	947	946
Decision Trees	931	567
Random Forests	630	634

5.2.2. Results Interpretation

In summary, the best-performing model is Random Forest with highest accuracy score and precision score, which did beat the benchmark set by theun-preprocessed method. In addition, the Random Forest model gives lower numbers of False Positives (FP) and False Negatives (FN).In our analysis, FP represents predicting a person who is looking for a job change, but actually they are not. FN represents predicting a person who is not looking for a job change, but actually they are. In the next part, we would further discuss the potential consequences of false predictions and suggested mitigation strategies.

6. Business Insights

6.1. Costs of False Predictions

6.1.1. Cost of FN

Predicting that a person is not looking for a job change when they are actually looking (a false negative) can have significant costs for the company who provides the training program.

Firstly, they will suffer talent loss. One critical aspect is the potential loss of valuable employees who are actively seeking job opportunities elsewhere. Failure to identify and address the needs of these individuals may result in the departure of key personnel possessing essential skills and experience. Additionally, there is a risk of losing specialized knowledge, skills, and expertise they acquired, contributing to a knowledge drain within the organization, affecting its overall competence and competitive edge.

Another encompassing problem for the company is the replacement costs. The process of recruiting and onboarding new employees incurs additional costs, including advertising, interviewing, and training expenses.

In addition, there are future impacts including skills gap and lack of succession planning. Losing employees with specific skills without adequate replacement can create a skills shortage, hindering the company's ability to meet business objectives. This will further result in a lack of succession development plan, making the company difficult to bloom in the industry.

6.1.2. Cost of FP

Predicting that a person is looking for a job change when they are not actually looking (a false positive) also has various costs for the company. Here are some of the potential costs:

Predicting a candidate's potential willingness to change jobs inaccurately can lead to the misallocation of crucial resources within a company. For instance, if the organization incorrectly anticipates that a candidate is likely to join, it might opt to invest in comprehensive and specialized training for that individual. This decision will result in wasted training costs. Moreover, the time and effort invested in monitoring the progress and evaluating the performance of candidates during training could further exacerbate the inefficiency of resource allocation. Additionally, recruitment costs are also wasted

False predictions will also affect the quality of training. If the company assumes that candidates are looking for a job change and tailors training programs accordingly, there might be a mismatch between the training content and the actual needs of the employees. This mismatch may result in training programs that are less effective and fail to address the genuine skill and

knowledge gaps within the workforce, ultimately impacting the overall competence and efficiency of the organization.

Unreliable predictions can have negtive effects on team dynamics. Making poor decisions based on inaccurate predictions may set a precedent for future decision-making processes within the company. This can erode the trust in leadership and undermine the organization's ability to make informed and strategic choices. Furthermore, misjudged predictions can lead to lower morale among existing employees who may feel uncertain about the stability of the team. The resulting sense of instability and insecurity can affect overall team performance and cohesion, posing challenges to the long-term success of the organization.

6.2. Methods to Mitigate the Cost

Mitigating the costs associated with false prediction is necessary. Here are some ways to mitigate these costs.

- Continuous Model Evaluation and Improvement

In order to improve the accuracy of the prediction model, the company should regularly assess the performance using real-world outcomes and implement a feedback loop to update and refine the model based on actual employee behavior and career decisions.

- Collect and Incorporate Employee Feedback

Another way to mitigate the cost is to gather feedback from employees about the training programs and the overall working environment. This process helps to gain insights into job satisfaction levels and corroborate the accuracy of our predictive models.

- Implement Secondary Screening Process such as Additional Assessments

What can also help is to introduce secondary screening processes such as additional assessments after training to evaluate the commitment and performance of newly trained employees. This allows the company to make more informed decisions before significant investments are made.